RCP Review

D.Bonham, E.Gallas, A.Jonckheere, J.Kowalkowski, H.Melanson, S.Veseli, S.White

1 Introduction

This review was termed a "code walkthrough/review". The review was to be focused on the Oracle database related code and the interfaces that exist to make this code possible to implement. It turns out to be difficult to comment directly on any one part of the code, given the large body of work that was presented at the meeting. The talks given by Marc and John did a good job at pointing the committee at some of the sources of the problems in the development of this package and what the current state is. Walking through the interfaces and some of the code was enough to get an initial understanding of the state and organization.

One of the goals of this review is to try to identify the reason that this project is taking so long to complete, given the initial time estimates, in order to prevent similar problems from appearing in the future. Another is to figure out how to correct these problems so that the project can be (1) made usable soon and (2) be maintainable. This brief report will concentrate on the problems we see in the development process and organization of the RCP database project, a plan for the short term, and a few recommendations for the longer term.

The fact that the RCP tests completed successfully has had a large impact on the detail and coverage of this review. The running tests indicate that the code discussed functions properly and it is believable that the system has hope of being exercised in the DØ environment in the very near future. More decisions may need to be made depending on near-term progress. This committee will not be disbanded until the progress reports outlined within this document are presented and further course of action is determined.

2 Problems

This project took much longer than expected and is way overdue. The developers and members of the review committee felt that the project should have been released in an alpha state in about two months. There are many things that contributed to the delays. The purpose of this section is to list some of the main problems that we found in the organization of this project in order to prevent future projects from falling into the same position. This list includes everything that we feel led to the long development of this project, some of which are just facts, and not necessarily problems.

- *Lack of experience with the technology.* Both Marc and John had little to no experience developing a CORBA application.
- *Two people with different schedules and priorities.* This arrangement is different than some of the database related projects, which only have one developer for both the client and server ends. This arrangement did not result in parallel work; it actually resulted in a large amount of wasted (idle) time between changes due to work on other active projects. There were other database projects with this same model, the difference being the developers worked much more closely together and had frequent contact with management.
- *CORBA exception mapping and handling (search for a general solution).* It seemed that a significant amount of time was dedicated to understanding the CORBA exception mechanism and trying to discover a good general way to manage exceptions from C++.
- *CORBA C++ Binding problems (search for a general solution).* The RCP system uses and presents classes that include standard library containers. The CORBA binding to C++ makes no use of the standard library so every data structure must be transformed into RCP C++ classes. A general templated translation was attempted to perform the translation. It could have been determined, after understanding the difficulties involved in creating the templated solution, that a simple brute-force solution may have been more appropriate for completing the alpha version of this product.

- *Iterations in the interface.* Early on in the project, the interface went through a series of iterations. This should be viewed as a natural part of the development process. The two-developer model may have caused more delays than necessary.
- *Movement of rules and logic from the client to the server.* Misunderstanding and miscommunications led to test failures and many iterations and corrections. The responsibilities within the client and server shifted as more was learned about CORBA and the database interactions.
- *Project schedule and regular required progress reports*. Lack of these can lead to projects going on forever. A review after the first missed milestone may have shortened the development cycle.
- *Different programming model than the current set of interoperating classes*. The RCP system was designed as a set of classes that communicate. The CORBA rules in use at DØ allow only synchronous communications, initiated by the client (get/put interface). This caused restructuring of the interface. It is unclear whether or not asynchronous communications (or bi-directional) would have helpful. Effects of synchronous use of CORBA can be seen in sam_manager, where a separate socket is opened to establish a way around the problem.
- *Poor communication with developers of similar applications.* Little effort was expended to leverage off the experience of SAM developers and developers of other database applications at DØ.

One lesson that can be learned here is that it is extremely important to **keep others on the project well informed of progress and problems, including management**. Others on the projects should be willing to listen and talk through different approaches to solving problems. Some of the communications problem can probably be attributed to differing design and development philosophies and strategies. It is easy to point fingers and try to identify a single source of the problem. This is not constructive and will not aid in the completion of this project at this point in time. In this case there appears to be many points of trouble, not all of which are directly caused by Marc and John. Inexperienced users may attribute some of the problems to an underdeveloped basic infrastructure for developing CORBA applications (this opinion is not shared by all committee members).

3 Recommendations

3.1 Regarding Schedule

The following subsections should be viewed as a timeline of activities that need to transpire. **It is highly recommended that Marc attend the database support meetings and report progress and problems**.

A well-organized and complete schedule should guide all future development activities. This schedule should be approved by management and frequently reviewed and updated as needed. The schedule should clearly state when versions of the product would be released. The dates assigned to the schedule should be taken seriously.

3.1.1 Preliminary Meeting

The question of short and long-term support of the python code in dbserver end of this project came up a number of times at this review. **A meeting of management needs to happen quickly to come to an understanding of support issues for this project**. It is recommended that this meeting occur 11/08 and include Patty McBride, SAM management, and Ruth Pordes. John's time and responsibilities for this project will also be discussed at this meeting.

3.1.2 Initial Step

Within one week of the completion and delivery of this document, we would like to see the RCP system up and running at DØ. In order for this to occur, the following constraints will be in place:

- *No addition features will be added*; the system will be used in its current form. It is understood that bulk loading will not be used.
- *No security will be in place*. It is understood that if the environment variables are changed to certain values, that a user can add or remove entries from the Oracle database in this version.

- *The exception handling in the client will remain in it current state.*
- *Performance will be ignored.*

John outlined many of the dbserver steps in the "Open Items" document; this list will not be reproduced here. To make this happen, some of things that must occur are listed below.

- *John must be available*. His help will likely be needed to deploy the dbserver and loading scripts.
- *Alan or Paul must be available*. Integration effort must be coordinated and carried out through them.
- *Scripts for loading* the database must be completed.
- *The **Integration database** must be loaded with RCPs.* This is the minimal level server that can be used for this initial deployment.
- *The official dbserver must be installed and functional*
- *A DDL code review must occur*. This includes the space report.
- *The user environment for connecting* with the dbserver must be established.

Having the alpha version available will allow measurements of RCP set retrieval speed to be collected. It will also allow the product to be tested in different modes than the RCP test suite. **John and Marc will report to this committee on the project status on a date to be established**. If the alpha version of this project has not been made available in the specified time period, then the course of action will need to be reevaluated at that time and will likely be significantly different then the schedule given below. If the system is running, but requires constant attention or is not operating as intended, then the alternate schedule may be needed.

3.1.3 Evaluation

Assuming that the alpha release is complete, the next step is to evaluate how well the system fulfills the needs of the experiment. This evaluation must initially include

- Measuring the time it takes to load an RCP set into a job.
- Get feedback from Alan and Paul

The findings should be reported to this committee and to management. Given that the client portion of this system and the dbserver portion have not been exercised under a typical load, **a simple load test program should be established**. This load test should be able to simulate concurrent reads of RCP sets that look like the farm starting and a continuous flow of analysis jobs coming and going. This program should give an initial guess as to what the worst case read time may be. In order to make good measurements, the dbserver end may need to distinguish the time to transfer the requested information to the client and the time it takes to accumulate the information from the database.

The committee members discussed the potential problems associated with the current design of one transaction per RCP requested in a job. It may be necessary to change this so that the entire RCP "tree" is transferred in one transaction. The timing values from the tests should give an indication whether or not this is necessary.

3.1.4 Near-term Development

Further development on this system depends heavily on the software support decision made by management and the availability of both Marc and John. **A single person must be responsible for both the client and server end**. This recommendation must be taken into account when deciding how to complete these activities. An outstanding task list should be produced to guide the work. Two items that must be corrected or completed are:

- Security
- Better exception handling

The timing results could easily increase this list to include:

- Transferring of the entire RCP "tree" in one query
- The capability of doing bulk updates/inserts

See the coding section of this document for recommendations on these exception handling. The trigger-db team has implemented security. **This team should be consulted to see if the solution used there also could be applied to this project**.

Development strategy and problems encountered in the previous steps may push these items into the "Future Efforts" category.

3.1.5 Future Efforts

If it is determined that any major development is needed to complete the alpha version of this project, then serious consideration should be given to immediately redoing the database interface, the database schema and simplifying both the client and server code. This course of action should also be considered if the system requires substantial fixes to make the alpha version work. **Please note that if changes such as the conversion to omniORBpy or adding a server side cache are not trivial, they should fall into this category**. If the alpha version of the project is successfully deployed, the course of action outlined here may need to be taken to make this product maintainable over a long period of time. The next version of the product (beta) could include all these changes. Marc or whoever will be maintaining this product will need to establish a schedule for completing this work and comment on its feasibility. Management will need to make a decision whether or not it makes sense to follow this course of action.

3.1.5.1 RCP Database Schema

The schema is based on requirements that are no longer relevant or can possibly be met by other means. This requirement was the ability to ask for RCP sets that contain parameters with specific names and values directly out of the database. The RCP object should be stored as a database CLOB (Character Large Object). This removes the need for all the scalar/vector tables and allows the dbserver end to treat the RCP object as a large chunk of text of which is knows nothing about. The effect should be the elimination of most of the dbserver code.

3.1.5.2 IDL Simplification

Given that the schema has changed according to the previous section, the interface can now be simplified. A good part of the development effort was devoted to transferring data structures from the server to the client that mimicked the C++ structures in the client. With the objects transferred as CLOBs, all the data structure definitions and manipulation code in the interface disappear.

3.2 Regarding Code

This section will be brief due to time constraints on preparing this document, the status of the project, and lack of time to go over the large body of code the project contains.

3.2.1 Client End

The exception processing needs to be improved. One example that was pointed out in the review session was attempting a reconnect when any non-std exception was caught and not distinguishing certain exceptions (user exception, connection failure, no data found). The SAM developers and trigger-db application developers indicated that they each have a way of dealing with exceptions on both the client and the server ends that works well for them. Marc should meet with the developers of these schemes. These schemes should be examined to see if they could be applied (in concept or implementation) to the RCP database-related classes. The results of these interactions should be presented at the regular database support meeting. Several committee members expressed concern that more communication with other developers was needed because this is a general problem that needs to be solved and others have solved it in a sensible way.

It was mentioned that the database support team and the trigger-db team also resolved this problem in an independent way, and that neither of the solution is perfect (but more complete that the RCP solution). Several committee members believe it would be beneficial to have one way of mapping exceptions used by all the various packages (where it makes sense) and that this should be pursued.

Any code that is replicated and changed from other packages, such as corba_util, should be fed back to the other packages. One example mentioned in the review was the singleton cleanup of a class in the corba_util package.

3.2.2 The Interface

The RCP database interface expressed in CORBA IDL appears to encompass several concepts. This feature made it difficult to grasp how the dbserver works and will likely cause maintenance problems in the future. It also removed the possibility of reusing some of the concepts in other packages. From what was presented at the review, it seems that one possible breakup of the interface can be:

- Get/Put user interface
- DB add/remove/bulk load administrative interface
- Authorization interface

The authorization interface could, for example, give out administrative objects or user interaction object depending on login information. Now the three interaction types can be tested and maintained separately.

3.2.3 Server End

Looking through the single, large python file is difficult. It would be nice to see a set of classes that work together to implement the interface, where each class does a specific part of the problem. This is a very vague statement and is meant to start a discussion about how one might design this dbserver so that it is more obvious where RCP rules are implemented and easier to verify that the code is correct – both visually and by test programs. One large class that has all the tables and rules visible in one place makes it difficult to determine if the code is correct, especially after a series of quick and dirty bug fixes are applied in the future.

The problem of exception handling was brought up here also. The database support team and the trigger-db team should be consulted to see if their exception mapping facility could be reused here.